TOKEN-LEVEL INTERPOLATION FOR CLASS-BASED LANGUAGE MODELS

Michael Levit, Andreas Stolcke, Shuangyu Chang, Sarangarajan Parthasarathy

Microsoft Corporation

ABSTRACT

We describe a method for interpolation of class-based n-gram language models. Our algorithm is an extension of the traditional EM-based approach that optimizes perplexity of the training set with respect to a collection of n-gram language models linearly combined in the probability space. However, unlike prior work, it naturally supports context-dependent interpolation for class-based LMs. In addition, the method works naturally with the recently introduced word-phrase-entity (WPE) language models that unify words, phrases and entities into a single statistical framework. Applied to the Calendar scenario of the Personal Assistant domain, our method achieved significant perplexity reduction and improved word error rates.

Index Terms— language model interpolation, class-based language models, context-dependent interpolation

1. INTRODUCTION

Language model interpolation is an essential part of state-of-the-art ASR. By building separate language models for different language domains, we acknowledge these domains' inherent diversity and postpone generating the final merged language model out of a number of pre-built components until more information is available about the target domain. For the currently prevalent linearly interpolated language models, the maximum-likelihood estimate of an n-gram probability is constructed as a linear combination of this n-gram's probabilities in all of the components, and the main interpolation challenge consists in finding such weights that optimize an agreed upon objective function. While there have been several successful attempts to relate objective functions directly to the word error rate metrics (especially, for second pass rescoring, e.g. [1, 2]), the de-facto standard remains perplexity-based optimization, largely due to its simplicity and scalability. For this selection of an objective function, the traditional solution is to use an expectation-maximization (EM) algorithm [3]. During the expectation stage, EM counts contributions of the individual components for the n-grams in the corpus, and in the maximization stage, it sets the re-estimated weight of a component to be its averaged contributions over the entire corpus.

While robust and easy to implement, the standard realization of the algorithm suffers from low resolution, forcing the components to retain the same interpolation weights in all circumstances. Several attempts to mitigate this shortcoming have been made in the past including count merging [4], Generalized Linear Interpolation [5] and history-dependent language model adaptation [6, 7]. The latter approach is particularly appealing since it requires very minimal change to the interpolation schema: instead of accumulating and averaging corpus-wide component contributions, the counting is performed on a per-n-gram-context basis. What is more, context-dependent interpolation weights can be easily utilized to build a standalone interpolated language model (as opposed to interpolation on-the-fly where no resultant merged LM needs to be stored at the end) because interpolation contexts correspond directly to the contexts of the n-gram probabilities.

In this paper, we focus on class-based language models that are known to improve generalization abilities of word LMs by supplementing training data with additional knowledge, and offer easy opportunities for adaptation and personalization. Specifically, we would like to extend EM to minimize perplexity of a joint language model with components containing classes. What makes interpolation of class-based LMs difficult is that there is no single shared linear word sequence to count over. In fact, each class-based component induces its own lattice of valid token-level parses[1], and the component-specific lattices often do not even share the same vocabulary (of tokens). One solution to bring the components to a common denominator is to project all parses back to the word level and use word prefixes to compute conditional word n-gram probabilities [8]. This approach is currently implemented in the SRILM toolkit [9]. While working well in many situations, this solution has two major drawbacks. First, it does not support context-dependent merging of the interpolated models into a single new model, because the word context is not explicitly encoded in the resulting merged class-based LM. Second, usability of per-word-context interpolation weights will be of little help if class definitions are dynamic or personalized because words from class definition in test/production might not even be present in the data used for training. Besides, just like the interpolation contexts are consistent with the probability contexts in the word-only case, we would like them to stay consistent for the class-based LMs. For instance, if sentence *"virginia smith lives in long island new york"* is parsed as *"NAME lives in CITY STATE"*, optimizing interpolation weights via word 5-gram *"virginia | smith lives in long"* is less intuitive, than optimizing them for token 5-gram *"PERSON | lives in CITY STATE"*.

The observations above suggest operating at the token level while interpolating class-based LMs. As we will show in the next sections, the issue of distinct parsing spaces is not a real obstacle and can be circumvented by exploring their (implicitly defined) union.

2. TOKEN-LEVEL LANGUAGE MODEL INTERPOLATION

In the course of the next sections, we will lean on notations previously introduced in [10] for corpus $\boldsymbol{W}$ of sentences $\boldsymbol{w}$ with normalized weights $L'(\boldsymbol{w})$ and alternative parses $\boldsymbol{c}^k$ in terms of tokens c_i^k that induce segmentations $\boldsymbol{\pi}^k = (\pi_1^k \dots \pi_I^k)$ of $\boldsymbol{w}$ (see Figure 1). In addition, for the purpose of LM interpolation, we need to introduce token-level history h_i^k that precedes c_i^k and a vector $\boldsymbol{\lambda}(h)$ of interpolation coefficients $\lambda_m(h)$ (one for each LM component) associated with this history.

Our first goal is to derive a formula for the log-likelihood of a data corpus in the interpolated LM. Given language model param-

[1] We will use the term "token" to subsume words, classes (and, potentially, word phrases).

Fig. 1. *Parsing sentences in terms of words, class and (possibly) common phrases.*

eters $\boldsymbol{\Theta}$ and a plurality $\boldsymbol{\Lambda}$ of interpolation weight vectors, this log-likelihood can be written as:

$$\log P(\boldsymbol{W}\,|\boldsymbol{\Lambda},\boldsymbol{\Theta}) = \sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \log P(\boldsymbol{w}\,|\boldsymbol{\Lambda},\boldsymbol{\Theta}) \tag{1}$$

For the sake of readability, from now on we will skip the explicit dependency on $\boldsymbol{\Lambda}$ and $\boldsymbol{\Theta}$. Next, suppose that we have already succeeded in building an interpolated language model. By construction, this model's token-level vocabulary will be defined as a union of the individual component vocabularies, and the class definitions will be shared as well. Applied to the individual sentences $\boldsymbol{w}$, this model will produce $K \geq 0$ alternative parses (token level representations) $\boldsymbol{c}^k$ with $k = \overline{1,K}$. For instance, the example sentence from Section 1 can be parsed as just the sequence of words it consists of, or as *"PERSON lives in CITY STATE"*, but also in many other ways (e.g. *"STATE smith lives in long island CITY"*). With this, our corpus likelihood can be written as:

$$\log P(\boldsymbol{W}) \quad = \quad \sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \log \left(\sum_{k} P(\boldsymbol{w}\,|\boldsymbol{c}^k) P(\boldsymbol{c}^k) \right) \tag{2}$$

where the two terms in the decomposition of $P(\boldsymbol{w})$ are:

1. probability of word-level realization $\boldsymbol{w}$ of the parse $\boldsymbol{c}^k$
2. language model probability of the parse according to the interpolated token-level LM.

With the help of the chain rule for LM probabilities and assuming statistical independence among word realization of individual tokens c_i^k, the likelihood turns into:

$$\sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \log \left(\sum_{k} \prod_{i} \left(P(\pi_i^k|c_i^k) P(c_i^k|h_i^k) \right) \right) \tag{3}$$

The terms $P(\pi_i^k|c_i^k)$ represent probabilities of various surface forms of a token. For instance, if the token is a simple one-of class, the term is the probability of the respective alternative.

Finally, let's recall that our LM is a product of linear interpolation of m LM components and substitute the corresponding expression into (3) obtaining the final formula for the corpus likelihood[2]:

$$\sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \log \left(\sum_{k} \prod_{i} P(\pi_i^k|c_i^k) \sum_{m} \left(\lambda_m(h_i^k) P_m(c_i^k|h_i^k) \right) \right) \tag{4}$$

Mind that the interpolation weights $\lambda_m(h_i^k)$ are already assumed context-dependent.

[2]For the sake of simplicity, we will use the same notation h for n-gram context and interpolation context; however, any practical implementation will need to functionally separate the two.

Our next goal is to optimize these weights so as to maximize the corpus (log)likelihood. Since this optimization is subject to stochastic conditions

$$\forall h : \sum_{m} \lambda_m(h) = 1, \tag{5}$$

optimal values for $\lambda_m(h)$ can be obtained by solving the following optimization problem:

$$\log P(\boldsymbol{W}) + \sum_{h} \gamma(h) \left(\sum_{m} \lambda_m(h) - 1 \right) \longrightarrow \max \tag{6}$$

where $\gamma(h)$ are Lagrangian coefficients (one for each context).

Next, we need to compute derivatives of the likelihood (4) over individual interpolation weights. After some simplifications that involve the general product derivative and Bayes rule formulae, we will arrive at:

$$\frac{\partial \log P(\boldsymbol{W})}{\partial \lambda_m(h)} \quad = \quad \sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \sum_{k:h\in\boldsymbol{c}^k} \left(P(\boldsymbol{c}^k|\,\boldsymbol{w}) \sum_{i:h_i^k=h} \frac{P_m(c_i^k|h)}{\sum_m P_m(c_i^k|h)\lambda_m(h)} \right) \tag{7}$$

Substituting this expression in (6) and solving w.r.t. $\lambda_m(h)$, we obtain:

$$\lambda_m(h) \quad = \quad -\frac{1}{\gamma(h)} \sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \sum_{k:h\in\boldsymbol{c}^k} \left(P(\boldsymbol{c}^k|\,\boldsymbol{w}) \sum_{i:h_i^k=h} S_{i,m}^k(\boldsymbol{w},h) \right) \tag{8}$$

where

$$S_{i,m}^k(\boldsymbol{w},h) \;:=\; \frac{P_m(c_i^k|h)\lambda_m(h)}{\sum_{m'} P_{m'}(c_i^k|h)\lambda_{m'}(h)} \tag{9}$$

is the contribution that a particular LM component has for a token in a given fixed history on a specific parse of a training sentence, and normalization coefficient

$$\gamma(h) = -\sum_{\boldsymbol{w}} L'(\boldsymbol{w}) \sum_{k:h\in\boldsymbol{c}^k} \left(P(\boldsymbol{c}^k|\,\boldsymbol{w})\; \#h|_{\boldsymbol{c}^k} \right) \tag{10}$$

is the expected number of times history h has been observed in the corpus[3]. Not surprisingly, the formula (8) receives a plausible interpretation as the average contribution of the m^{th} component for the tokens that occur following context h.

To make (8) actionable, we still need to offer a way to compute posterior probability of a parse $P(\boldsymbol{c}^k|\,\boldsymbol{w})$. This can be accomplished with Bayes rule:

$$P(\boldsymbol{c}^k|\,\boldsymbol{w}) = \frac{P(\boldsymbol{w},\boldsymbol{c}^k)}{\sum_{\boldsymbol{c}} P(\boldsymbol{w},\boldsymbol{c})} \tag{11}$$

where joint probabilities $P(\boldsymbol{w},\boldsymbol{c})$ are computed as

$$P(\boldsymbol{w},\boldsymbol{c}) = \prod_{c_i\in\boldsymbol{c}} P(c_i|h_i)P(\pi_i|c_i). \tag{12}$$

While $P(\pi_i|c_i)$ are within-class probabilities that are fixed (within each iteration), the language model probability for the interpolation

[3]We use notation $\#\cdot|_{\boldsymbol{c}}$ to signify counting along parse $\boldsymbol{c}$.

scenario can be further re-expressed in terms of per-component probabilities, leading to:

$$P(\boldsymbol{w},\boldsymbol{c}) = \prod_{c_i \in \boldsymbol{c}} \left(P(\pi_i|c_i) \sum_m \lambda_m(h_i) P_m(c_i|h_i) \right). \qquad (13)$$

Formulae (8-10) offer themselves nicely for optimization via EM. During the expectation step, for each context h we accumulate weighted contributions along every parse $\boldsymbol{c}^k$ that has this context, and in the maximization step, the ML-estimates for weights $\lambda_m(h)$ are obtained via normalization.

3. EXAMPLE

To illustrate differences between word-based and token-based interpolation approaches, consider the example of two language models. The first one is word-based (no classes), while the other one has a single class of city names C which is defined as a set of weighted alternatives, including *"new york"* with weight μ. Let the probability of unseen words in each LM be 0. Next, suppose that we have a training sentence "$<s>\, new\ york <\!/s>$" to contribute sufficient statistics like (9) for context "$<s>\, new$".

For the word-based approach, conditional probabilities $P_m(york|<s>\, new)$ for $m = 1, 2$ need to be estimated via word prefix probabilities [8]:

$$P_m(york|<s>\, new) = \frac{P_m^{\text{pref}}(<s>\, new\ york)}{P_m^{\text{pref}}(<s>\, new)}.$$

To compute a prefix probability, we consider all possible derivations starting with this prefix and sum up their forward probabilities [8]. Each of these forward probabilities can be decomposed into a number of conditionals. For the sake of brevity, let us focus on the numerator (denominator is computed similarly). In the first LM, a single derivation is possible with all conditionals coming directly from the LM n-grams:

$$\begin{aligned} P_1^{\text{pref}}(<s>\, new\ york) \quad &= \\ P_1^{\text{LM}}(<s>)\, P_1^{\text{LM}}(new|<s>)\, P_1^{\text{LM}}(york|<s>\, new). & \end{aligned}$$

In the second model, two derivations exist: one is word-only, the other goes through the class C:

$$\begin{aligned} P_2^{\text{pref}}(<s>\, new\ york) \quad &= \\ P_2^{\text{LM}}(<s>)\, P_2^{\text{LM}}(new|<s>)\, P_2^{\text{LM}}(york|<s>\, new) \quad &+ \\ P_2^{\text{LM}}(<s>)\, P_2^{\text{LM}}(C|<s>) \times & \\ P_2^{\text{class}}(new\ .\ york|C)\, P_2^{\text{class}}(york|new\ .\ york, C) & \end{aligned}$$

with the "."-notation denoting a partially expanded nonterminal, as used in the Earley parsing framework [8]. Due to our simple class definition, the second summand in this formula turns into $\mu\, P_2^{\text{LM}}(<s>)\, P_2^{\text{LM}}(C|<s>)$.

The token-based approach is principally different in that it does not include alternative implicit derivations of a sentence prefix in the probability computation, but rather considers a number of explicit parses of the sentence obtained with the previous LM combination, and their corresponding posteriors. Suppose these parses are: $\boldsymbol{c}^{(1)}$ ="$<s>\, new\ york <\!/s>$" and $\boldsymbol{c}^{(2)}$ ="$<s>\, C <\!/s>$" with respective posteriors $\nu^{(1)}$ and $\nu^{(2)}$. The conditional probabilities $P_m(york|<s>\, new)$ needed for sufficient statistics for context "$<s>\, new$" will now be borrowed directly from the language models (with the appropriate smoothing); however, the contributions will be weighted by $\nu^{(1)}$ since this context only occurs in the first parse $\boldsymbol{c}^{(1)}$. This approach allows for interpolation weights to be estimated also for contexts containing classes (such as "$<s>\, C$"). The produced weights do not depend on the specific class instances, which means that several LMs can be merged into a single LM in a context-dependent manner with maximum likelihood merged probability estimates computed as

$$\begin{aligned} P(\cdot|<s>\, C) \quad &= \\ \lambda_1(<s>\, C) P_1(\cdot|<s>\, C) + \lambda_2(<s>\, C) P_2(\cdot|<s>\, C) & \end{aligned}$$

4. DISCUSSION

A few words need to be said to justify the validity of the approach. Initially, we postulated that the interpolated language model is already available to us. In reality, of course, it is not, and the interpolation (13) might force some components to provide probabilities for a parse that contains tokens outside of their respective vocabularies. However, such tokens can always be mapped to dedicated "unknown" symbols, and most language models already contain estimated probabilities of unknowns in them. If not, these probabilities can be considered zero and easily absorbed in the computation. Therefore, our algorithm does not violate probabilistic constraints.

On a separate note, as the above formulae explicitly refer to individual parses, a simple extension of the algorithm can deal with compact representations of the entire parsing space (lattices), in which case instead of conditioning on probabilities of entire end-to-end parses, we condition on individual arcs in the graph, using forward and backward probabilities. For the sake of the present study, however, we are focusing on 10 highest scoring parses.

As for context-dependent interpolation, the straightforward look-up table implementation is prone to over-training, since it produces separate vectors of interpolation weights for each history, no matter how often this history appeared in training. To produce robust estimates, a number of techniques can be employed such as hierarchies of contexts and MAP estimation [11]. For our experiments, we found a simple combination of hierarchical contexts with absolute thresholds on the minimum number (typically between 3 and 5) of context observations to work the best.

Finally, it should be noted that the approach presented is not limited to traditional class-based LMs, but can as well be applied to modifications such as word-phrase-entity (WPE) LMs [10] that build language models from corpora expressed in terms of words, classes and phrases in a way that optimizes their likelihood.

5. EXPERIMENTS AND RESULTS

We selected the Personal Assistant, Calendar scenario for our experiments. The target domain is a class of utterances that have been *automatically classified* to pertain to the Calendar scenario for which we have two sets: TD1 (to train interpolation weights) and TD2 (to test). Note that the prediction accuracy is relatively low and there are a significant number of irrelevant samples in TD1 and TD2 alike. There are two LM components to interpolate. The first one (LM1) is a general purpose language model. The second one (LM2) is a 3-gram LM to be built from a separate collection of sentences VD1 that were selected by human labelers as coming from the Calendar scenario. Table 1 summarizes the characteristics of these datasets.

Corpus	LM	Properties
TD1 (test)	-	3K sent., 20K words
TD2 (weight training)	-	10K sent., 67K words
VD1 (close to target domain)	LM2 word/class	20K sent. 176K words; 3-gram, 5K lexicon
generic	LM1	5-gram LM, 50K lexicon

Table 1. *Corpora and data LM at a glance.*

The second component LM2 can be trained as a regular word-level n-gram language model (LM2-word) or as a class-based language model (LM2-class). In the experiments below a special case of class-based models, the WPE LM [10], is employed for this purpose. The definition of the six classes (Time, Date, Weekdays, FirstName, State, City) is also borrowed from [10] with some of the classes being defined as word tries, and others encoded as finite state automata. Table 2 shows perplexity and word error rates achieved on the test set TD1 by various training configurations. For the purpose of parsing, we postulated $P_m(\text{OOV}) = 0, \forall m$, and in the context-dependent case, required a context to have expected number of observations of at least 3.0 to generate its own vector of interpolation weights.

LangMod	ppx	word-oov	WER
LM1	104.7	0.72%	12.22%
LM2-word	77.8	7.28%	22.34%
LM2-word + LM1	72.2	0.68%	11.31%
LM2-word + LM1 (CD)	66.8	0.68%	11.14%
LM2-class	72.9	6.80%	21.20%
LM2-class + LM1	69.6	0.68%	10.93%
LM2-class + LM1 (CD)	63.0	0.68%	10.78%

Table 2. *Evaluation results with language models constructed in various ways.*

First of all, we see that the WPE LM (LM2-class) achieves better perplexity and WER relative to the word-level LM (LM-word) trained on the same data. In combination with the generic model LM1 that offers better overall WER (perplexities are not directly comparable due to significant differences in the OOV rates), all metrics are improved. The advantage of the WPE LMs carry over to the interpolated models as well. Finally, interpolating with context-dependent weights (CD) helps both combinations equally, resulting in significant perplexity drops of about 10% for LM combinations with LM2-word as well as LM2-class, but also reducing WER. Overall, the difference between context-independent interpolation with word-level LM and context-dependent interpolation with WPE LM amounts to almost 13% perplexity reduction and statistically significant 4.7% WERR.

In a slightly different setup, we removed phrases from WPE definitions and only kept classes that can be represented as weighted lists of items. This configuration enabled us to run a side-by-side comparison between token-based interpolation with the word-based interpolation (as implemented in the SRILM [9]) for class-based language models. The token-level context-independent interpolation resulted in perplexity 71.3, slightly better than 72.1 of the word-level interpolation. We did not perform the comparison for a context-dependent version because for this case it is not possible to create a single merged model (which is one of the advantages of token-level interpolation).

A possible objection to context-dependent interpolation is that it requires more in-domain data than context-independent interpolation, as more parameters need to be trained [11]. Instead of using 10K examples of the TD2 corpus, maybe it would be more beneficial to add them to the training material for LM2. To investigate how to best use available data, we started chipping off chunks of different sizes from TD2, adding them to VD1 instead. The rest was still used to train interpolation weights. While WPE LM training algorithm allows for entity adaptation (within-class weight optimization) in parallel to n-gram probability estimation [10], for our next experiment we did not do it and focused solely on LM training. The plot in Figure 2 demonstrates how different data splits affect overall perplexity of the resultant interpolation language model for word- and WPE LMs. The x-axis shows the split of the data in TD2 between additional LM2 training and weight-training material. For instance, "9000-1219" means that out of the total of 10219 TD2 sentences, 9000 were added to LM2 to train in-domain class-based LM and the remaining 1219 were used for context-dependent weight optimization.

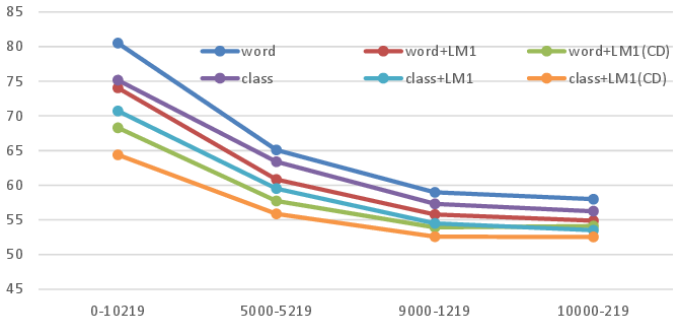

Fig. 2. *Effect of moving training samples from weight training to LM training set on test set perplexity.*

While direct use of in-domain data as LM training material appears to have higher impact on final LM quality, the improvements due to class-based WPE LM and context-dependent interpolation remain consistent. Even with slightly more than 200 training examples, context-dependent weights yield 5% perplexity reduction, most likely due to the separate weights for the very common begin-of-sentence context. In a separate experiment, we used the entire corpus TD2 LM *and* weight training, achieving small additional improvements relative to the "10000-219" combination. Practical considerations might still dictate the use of additional training data for interpolation weights, i.e., when it is not convenient to modify pre-existing component LMs.

6. CONCLUSIONS AND FUTURE WORK

We have presented a novel method for token-level interpolation of class-based language models. Our algorithm modifies the traditional interpolation weight estimation approach by letting it count statistics from alternative parses produced for each training sentence by a linear combination of component language models. While the results of context-independent interpolation exhibit similar qualities as standard word-level probability interpolation, our method allows for a straightforward extension to handle context-dependent interpolation weights, and in this way, achieves better perplexity and recognition results. We applied the method to Calendar scenario in the Personal Assistant domain and observed significant perplexity and WER reductions w.r.t. the single-LM baselines, but also relative to context-independent interpolation.

7. REFERENCES

[1] P. Beyerlein, "Discriminative model combination," in *Proc. ICASSP*, Seattle, WA, May 1998.

[2] A. Deoras, D. Filimonov, M. Harper, and F. Jelinek, "Model combination for speech recognition using empirical Bayes risk minimization," in *Proc. SLT*, Berkeley, CA, Dec 2010.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the em algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[4] M. Bacchiani and B. Roark, "Unsupervised language model adaptation," in *Proc. ICASSP*, Brisbane, Australia, April 2003, pp. 224–227.

[5] B. Hsu, "Generalized linear interpolation of language models," in *Proc. ASRU*, Kyoto, Japan, Dec 2007, pp. 136–140.

[6] I. Bulyko, M. Ostendorf, and A. Stolcke, "Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures," in *Proc. HLT-NAACL*, M. Hearst and M. Ostendorf, Eds., Edmonton, Alberta, Canada, Mar. 2003.

[7] X. Liu, M. J. F. Gales, and P. C. Woodland, "Context dependent language model adaptation," in *Proc. Interspeech*, Brisbane, Australia, 2008.

[8] A. Stolcke, "An efficient probabilistic context-free parsing algorithm that computes prefix probabilities," *Computational Linguistics*, vol. 21, pp. 165–201, Jun 1995.

[9] A. Stolcke, "SRILM—an extensible language modeling toolkit," in *Proc. ICSLP*, Denver, CO, 2002, pp. 901–904.

[10] M. Levit, S. Parthasarathy, S. Chang, A. Stolcke, and B. Dumoulin, "Word-phrase-entity language models: Getting more mileage out of n-grams," in *Proc. Interspeech*, Singapore, September 2014, ISCA - International Speech Communication Association.

[11] X. Liu, M. J. F. Gales, and P. C. Woodland, "Use of contexts in language model interpolation and adaptation," *Computer Speech and Language*, vol. 27, pp. 301–321, Jan 2013.